Pollux
Manuel d'utilisation

Frédéric Mathiot - Vincent Maurin

Table des matières

Installation 2
- Matériels et logiciels requis 2
- Déploiement 2
- Configuration 3
- Lancement 3

Gestion du manager 4
- PolluxManagerGUI 4
  - Pré-requis 4
  - Gestion des managers 4
  - Gestion des contrôleurs 6

Écriture de contrats et de scripts 7
- Un exemple pas à pas 7
- Description de l'API 10
  - PolluxMain 11
  - IManager 11
  - IObservableControlleur 11
  - Observer 13
  - IViolationStrategy 13
  - ViolationStrategy 13

Installation

Matériels et logiciels requis

Un serveur sur lequel sont installés

- Un serveur Web compatible .Net Frameworks (IIS)
- Microsoft .NET Frameworks 2.0

Une ou plusieurs machines clients pour exécuter les contrats avec

- C# script (http://www.members.optusnet.com.au/~olegshilo/index.html)
- Microsoft .NET Frameworks 2.0

Déploiement

Sur le serveur :

- PolluxWebLibrary, fichier PolluxWebLibrary.dll à placer dans le répertoire bin du serveur web et dans le répertoire de l'application
- PolluxProgram, executable
- PolluxProxyGenerator, fichier PolluxProxyGenerator.dll à placer dans le répertoire de l'application
- PolluxLibrary, fichier Pollux.dll à placer dans le répertoire bin du serveur web et dans le répertoire de l'application
- pollux.xml, fichier de configuration à placer dans le répertoire de l'application

Exemple

```
|- C:\
|--- www\ (répertoire correspondant à un répertoire virtuel de IIS)
|----- bin\
|------- Pollux.dll
|------- PolluxWebLibrary.dll
|--- Pollux\
|----- PolluxProgram.exe
|----- Pollux.dll
|----- PolluxWebLibrary.dll
|----- PolluxProxyGenerator.dll
|----- pollux.xml
```

Sur les clients :

- PolluxLibrary, fichier Pollux.dll à placer dans le répertoire du C# script
- PolluxManagerGUI, interface graphique pour gérer des managers

Configuration

Il faut régler plusieurs paramètres de PolluxProgram (fichier pollux.xml):

- le chemin du répertoire correspondant à la racine du serveur web
- le port utilisé par le manager

Exemple

```
<?xml version="1.0"?>
<PolluxConfiguration                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Port>7000</Port>
  <WebServerRootPath>c:\www</WebServerRootPath>
</PolluxConfiguration>
```

Lancement

L'exécutable de PolluxProgram doit être lancé. Celui-ci crée et initialise le manager. Il peut être éventuellement lancé au démarrage ou en tant que service.

Gestion du manager

Une fois le manager lancé, deux possibilités sont offertes pour le gérer, grâce à l'API utilisée dans des scripts (voir section suivante) ou grâce à PolluxManagerGUI.

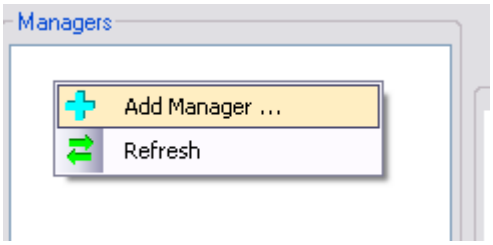
PolluxManagerGUI

Pré-requis

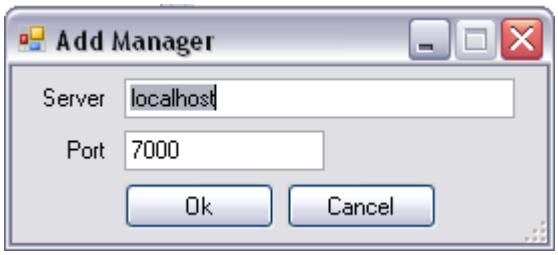
- .NET Frameworks 2.0
- Le répertoire où se trouve l'exécutable PolluxManagerGUI doit contenir Pollux.dll

Gestion des managers

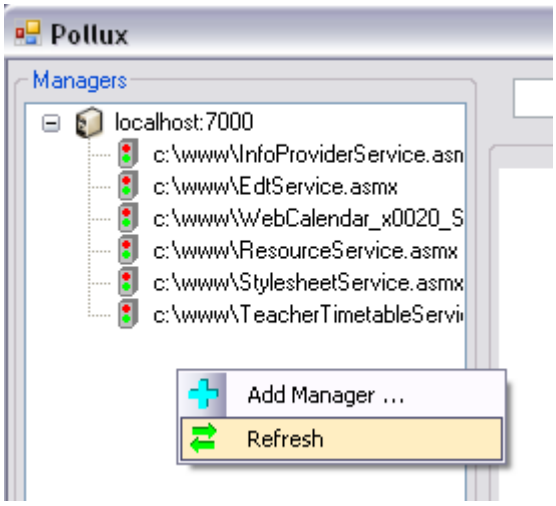
Pour gérer un manager, il faut l'ajouter dans le panneau gauche. Pour cela, il suffit de cliquer avec le bouton droit dans ce panneau et sélectionner 'Add Manager'.



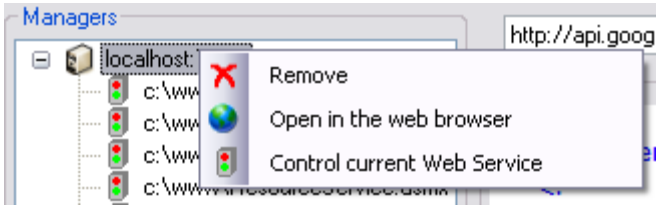
Une boite de dialogue s'ouvre alors où l'utilisateur précise l'adresse et le numéro de port du manager.



Une fois ajouté, le manager apparaît dans la liste, ainsi que les contrôleurs associés. L'utilisateur a aussi la possibilité de rafraîchir ce panneau avec un clic droit et 'Refresh'.



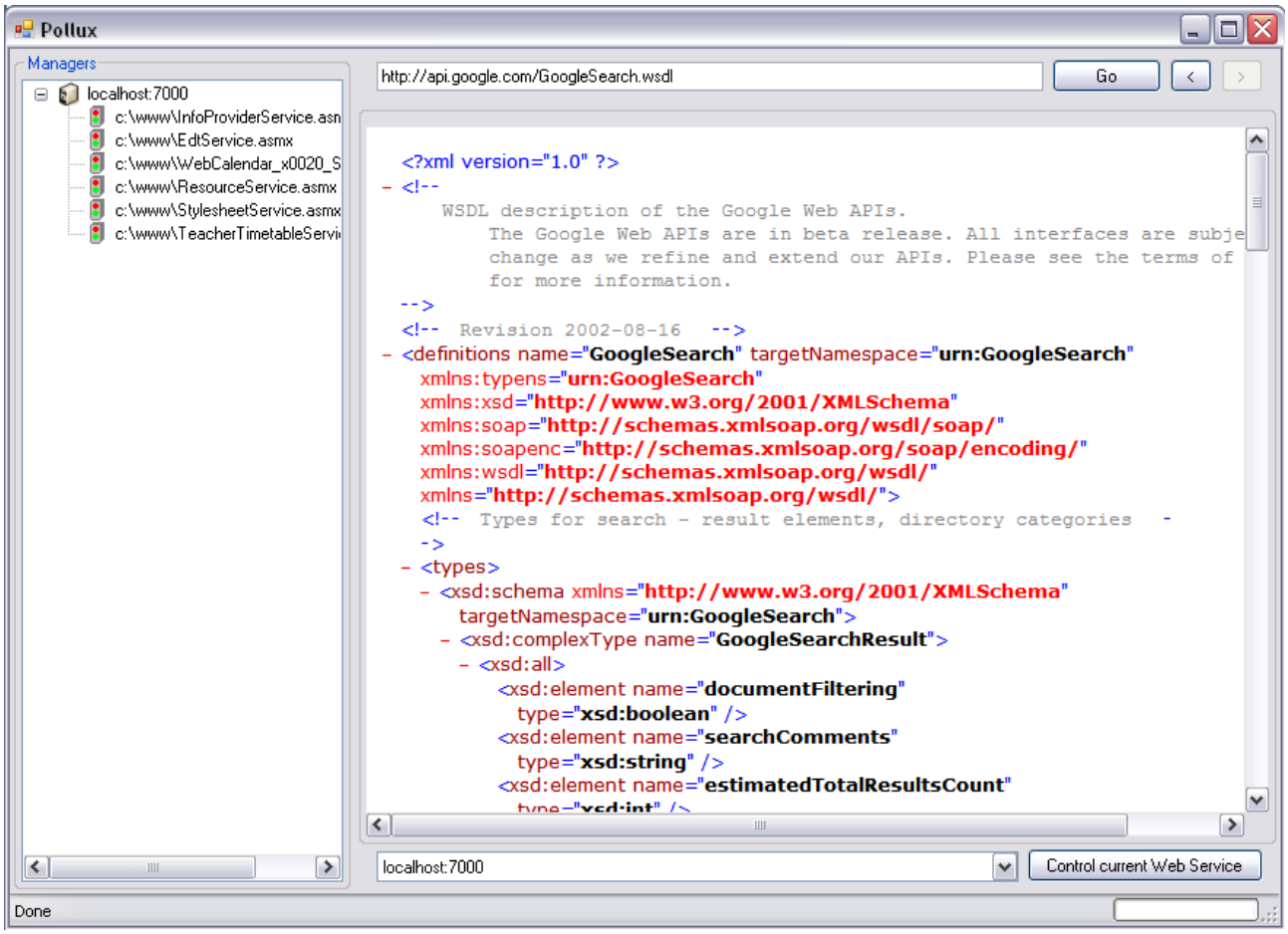
Une fois le manager ajouté, certaines fonctionnalités sont accessibles par un clic droit.



'Remove' pour le supprimer de la liste.
'Open in the Web browser' pour naviguer sur ce serveur grâce au panneau droit.
'Control current web service' voir Gestion des Contrôleurs.

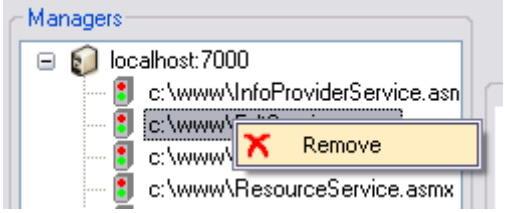
Gestion des contrôleurs

La panneau de droite est navigateur simplifié permettant d'accéder à des serveurs Web. Il permet ainsi d'accéder au description de Web Service (WSDL). Il est possible de créer un contrôleur pour le WSDL ouvert.



Pour créer un contrôleur sur le Web Service ouvert dans le navigateur, il faut soit accéder au menu contextuel d'un manager (clic droit sur le manager), soit sélectionner un manager dans la liste et cliquer sur 'Control Current Web Service'.

Les contrôleurs existants sur un manager peuvent être supprimés grâce au menu contextuel (clic droit sur le contrôleur)



Un exemple pas à pas

Un Web Service 'Répertoire' offre un service de gestion de contact. Un contact est défni par un nom, un prénom et un email. Une méthode 'ajouterContact' prend un objet de type 'Contact' en paramètre. Notre contrat va vérifier que les champs du type 'Contact' ont une syntaxe correcte.
Le principe est d'inscrire un observeur en pre-appel, qui vérifie le paramètre d'appel.
Ici, le paramètre est de type 'Contact', un type complexe. Pour garantir une transmission correcte, tous les objets transmis aux observeurs sont sérialisés en XML. Il nous faut donc avoir une classe 'Contact' définie, pour pouvoir la désérialiser. Soit nous avons accès aux classes du Web Service, soit il faut définir nous même notre propre classe 'Contact'.

Elle doit avoir les mêmes champs, avec le même nom. Il faut aussi préciser l'espace de nom XML (celui du Web Service), pour pouvoir désérialiser correctement.

```
[System.Xml.Serialization.XmlTypeAttribute(Namespace = "http://www.repertoire.essi.fr")]
public class Contact
{
    private string _nom;

    public string Nom
    {
        get { return _nom; }
        set { _nom = value; }
    }

    private string _prenom;

    public string Prenom
    {
        get { return _prenom; }
        set { _prenom = value; }
    }

    private string _email;

    public string Email
    {
        get { return _email; }
        set { _email = value; }
    }
}
```

L'observeur, qui sera inscrit en pre-appel, va donc recevoir un PreCallEvent contenant le paramètre de la méthode. Il faut qu'il le désérialise puis vérifie les champs.

```
public class MyObserver : Observer
{
    public override bool callHandler(CallEvent callEvent)
    {
        // Deserialisation du parametre
               Contact  contact  =  (Contact)  new  XmlSerializer(typeof(Contact)).Deserialize(new
System.IO.StringReader (((PreCallEvent)callEvent).Parameters[0]));

        // Verification (la méthode contactInvalide n'est pas codée ici,
        // elle fait simplement des vérifications sur des chaînes)
        if (contactInvalide (contact))
        {
             // La contrat est violé, on fait appel à la stratégie par défaut
             // (exception au client)
             return ViolationStrategy.getDefaultStrategy().execute(null);
        }
        return true;
    }
}
```

La méthode Main du contrat doit récupérer une référence vers le contrôleur (grâce au manager) puis y inscrire l'observeur.

```
public static void Main()
{
    // Recuperation de la reference vers le manager
    IManager manager = PolluxMain.getManager("tcp://localhost:7000/Manager");

    // Recuperation du contrôleur
    IObservableController controller =
                  manager.getObservableController("c:\\www\\Repertoire.asmx");

    // Inscription sur ajouterContact en pre-appel et en mode bloquant
    controller.subscribePreCall(new MyObserver(), "void ajouterContact (Contact)", true);

    Console.WriteLine("Ok");
    Console.ReadLine();
}
```

Au final, voici le code obtenu :

```
using System;
using Pollux;
using System.Xml;
using System.Xml.Serialization;
using System.IO;

namespace Contrat
{
    [System.Xml.Serialization.XmlTypeAttribute(Namespace = "http://www.repertoire.essi.fr")]
    public class Contact
    {
        private string _nom;
        public string Nom
        {
            get { return _nom; }
            set { _nom = value; }
        }
        private string _prenom;
        public string Prenom
        {
            get { return _prenom; }
            set { _prenom = value; }
        }
        private string _email;
        public string Email
        {
            get { return _email; }
            set { _email = value; }
        }
    }

    public class RepertoireContrat
    {
        public class MyObserver : Observer
        {
            public override bool callHandler(CallEvent callEvent)
            {
                // Deserialisation du parametre
                  Contact contact = (Contact) new XmlSerializer(typeof(Contact)).Deserialize(new
System.IO.StringReader (((PreCallEvent)callEvent).Parameters[0]));

                    // Verification (la méthode contactInvalide n'est pas codée ici, elle fait
simplement des vérifications sur des chaînes)
                if (contactInvalide (contact))
                {
                     // La contrat est violé, on fait appel à la stratégie par défaut (exception
au client)
                    return ViolationStrategy.getDefaultStrategy().execute(null);
                }
                return true;
            }
        }

        public static void Main()
        {
            // Recuperation de la reference vers le manager
            IManager manager = PolluxMain.getManager("tcp://localhost:7000/Manager");

            // Recuperation du contrôleur
                                                    IObservableController    controller    =
manager.getObservableController("c:\\www\\Repertoire.asmx");
            // Inscription sur ajouterContact en pre-appel et en mode bloquant
            controller.subscribePreCall(new MyObserver(), "void ajouterContact (Contact)", true);
            Console.WriteLine("Ok");
            Console.ReadLine();
        }
    }
}
```

Pour exécuter notre contrat, il suffit, avec le c# script de lancer la commande suivante (notre contrat se trouve dans RepertoireContrat.cs)

```
> cscs.exe RepertoireContrat.cs
```

Description de l'API

PolluxMain

- ➔ public static IManager getManager(string location)

Récupère le manager grâce à son adresse.

```
IManager manager = getManager ("tpc://127.0.0.1:7000/Manager");
```

IManager

- ➔ string[] getManagedWebServices()

Récupère la liste des Web Services managés. Les Web Services sont indexés par le chemin absolu du fichier asmx sur le serveur web

```
string [] controllerPathes = getManagedWebService();
```

- ➔ string createProxyController(string wsdlUri)

Crée un couple proxy/contrôleur pour un Web Service. L'url de son WSDL doit être passé en paramètre.

```
string controllerPath = createProxyController ("http://www.foo.com/service.asmx?WSDL
```

- ➔ IObservableController getObservableController(string webServiceFile)

Récupère une référence vers un contrôleur avec l'index vers un Web Service (récupéré grâce à getManagedWebServices)

```
IObservableController controller = getObservableController ("c:\\www\\Service.asmx");
```

- ➔ void removeObservalbeController(string webServiceFile)

Supprime le contrôleur et le proxy pour un Web Service donné (toujours grâce à son index).

```
removeObservableController ("c:\\www\\Service.asmx");
```

IObservableControlleur

- ➔ void subscribePreCall(Observer observer, string methodSignature, bool isBlocking)

Enregistre un observeur auprès du contrôleur en pré-appel. Deux paramètres en plus de la référence vers l'observeur sont requis. Une signature de méthode, donnée sous la forme d'un expression régulière, qui sera comparée avec la signature des méthodes du Web Service, et un booléen indiquant si le contrôleur attend une réponse ou non de l'observeur.

```
controller.subscribePreCall (myObserver, ".*method.*", true);
```

➔ void subscribePostCall(Observer observer, string methodSignature, bool isBlocking)

Enregistre un observeur auprès du contrôleur en post-appel. Le fonctionnement est similaire à subscribePreCall.

```
controller.subscribePostCall (myObserver, ".*method.*", true);
```

➔ void unsubscribePreCall(Observer observer)

Désenregistre un observeur en pré-appel.

```
controller.unsubscribePreCall (myObserver);
```

➔ void unsubscribePostCall(Observer observer)

Désenregistre un observeur en post-appel.

```
controller.unsubscribePostCall (myObserver);
```

Observer

➔ public abstract bool callHandler(CallEvent callEvent)

Cette méthode doit être implémentée par tous les observeurs. Elle reçoit un PreCallEvent ou un PostCallEvent selon si l'observeur est enregistré en pré ou post-appel.
Elle doit retourner un booléen, soit directement, soit en faisant appel à une stratégie de violation. La valeur retournée est uniquement pris en compte si l'observeur est enregistré en mode bloquant. Si le retour est faux, une exception sera lancée au client.

```
public class PreCallHandler : Observer
{
      public override bool callHandler(CallEvent callEvent)
      {
            Console.WriteLine("Methode appelee : " + callEvent.MethodName + " sur le service " +
callEvent.ServiceName);
            return true;
      }
```

IViolationStrategy

➔ public bool execute (Dictionnary<string,object> parameters)

Cette méthode exécute une stratégie en lui fournissant les paramètres nécessaires dans un dictionnaire.

```
Dictionary <string,Object> dictionnary = new Dictionary <string,Object>();
dictionnary["MSG"] = "Taille Superieure a 800";

return ViolationStrategy.getStrategy(StrategyType.Trace).execute(dictionnary);
```

ViolationStrategy

➔ public IViolationStrategy getDefaultStrategy ()

Retourne la stratégie par défaut.

```
return ViolationStrategy.getDefaultStrategy().execute(null);
```

➔ public IViolationStrategy getStrategy (StrategyType type)

Retourne une stratégie désignée par son type.

```
return ViolationStrategy.getStrategy(StrategyType.Basic).execute(null);
```